Bounded-Rate Multi-Mode Systems Based Motion Planning

Devendra Bhave
IIT Bombay
devendra@cse.iitb.ac.in

Sagar Jha
IIT Bombay
sagarjha@cse.iitb.ac.in

Shankara Narayanan Krishna
IIT Bombay
krishnas@cse.iitb.ac.in

Sven Schewe
University of Liverpool
sven.schewe@liverpool.ac.uk

Ashutosh Trivedi
IIT Bombay
trivedi@cse.iitb.ac.in

ABSTRACT

Bounded-rate multi-mode systems are hybrid systems that can switch among a finite set of modes. Its dynamics is specified by a finite number of real-valued variables with mode-dependent rates that can vary within given bounded sets. Given an arbitrary piecewise linear trajectory, we study the problem of following the trajectory with arbitrary precision, using motion primitives given as bounded-rate multi-mode systems. We give an algorithm to solve the problem and show that the problem is co-NP complete. We further prove that the problem can be solved in polynomial time for multi-mode systems with fixed dimension. We study the problem with dwell-time requirement and show the decidability of the problem under certain positivity restriction on the rate vectors. Finally, we show that introducing structure to the multi-mode systems leads to undecidability, even when using only a single clock variable.

Categories and Subject Descriptors

D.4.7 [**Organization and Design**]: Real-time systems and embedded systems; B.5.2 [**Design Aids**]: Verification

General Terms

Theory, Verification

Keywords

Switched Systems, Motion Planning, Hybrid Automata

1. INTRODUCTION

Hybrid automata [2] are a natural and expressive formalism to model systems that exhibit both discrete and continuous behavior. Intuitively, hybrid automata extend the discrete system modeling framework of extended finite state machines with continuous variables modeled along continuous dynamical systems such that the flow of continuous variables in each state is modeled as a system of first-order ordinary differential equations. Discrete jumps in the values of the variables are modeled via resets on the transitions of the automata. However, the applications of hybrid automata in analyzing cyber-physical systems have been rather limited due to undecidability [9] of simple verification problems such as reachability. This drawback of hybrid automata has fueled the investigation of the so-called compositional methodology [8, 12] to design complex system by sequentially composing well-understood lower-level components. This methodology has, for example, been used in the context of the *motion planning* problem for mobile robots, where the task is to move a robot along a pre-specified trajectory with arbitrary precision by sequentially composing a set of well-studied simple motion primitives, such as "move left", "move right" and "go straight". In this paper, we investigate the motion planning problem for systems, whose motion primitives are given as constant-rate vectors with uncertainties.

We consider bounded-rate multi-mode systems [4] that can be considered as *constant-rate multi-mode systems [5] with uncertainties.* These systems consist of a finite set of continuous variables, whose dynamics is given by mode-dependent constant-rates that can vary within given bounded sets. In such systems, the dynamics of the system can be viewed as a two-player game between a controller and the environment. In each step, the controller chooses a mode and time duration and the environment chooses a rate vector for that mode from the given bounded set. The system evolves with that rate for the chosen time. The game continues in this fashion from the resulting state. Alur, Trivedi, and Wojtczak [5] considered constant-rate multi-mode systems and showed that the reachability problem—deciding the reachability of a specified state while staying in a given safety set—and the schedulability problem—deciding the existence of a non-Zeno control so that the system always stays in a given bounded and convex safety set—for this class of systems can be solved in polynomial time. Alur et al. [4] showed that the existence of robust control for the schedulability problem for bounded-rate multi-mode systems is, although intractable (co-NP-complete), decidable. However, they left the decidability of the robust reachability problem for this class of systems open.

The robust reachability problem for bounded-rate multi-mode system is defined as follows: given a bounded-rate multi-mode system, a starting state, and a target state, decide whether it is possible to reach the target state from the starting state with arbitrary precision. The key result of this paper is the decidability of the robust reachability problem

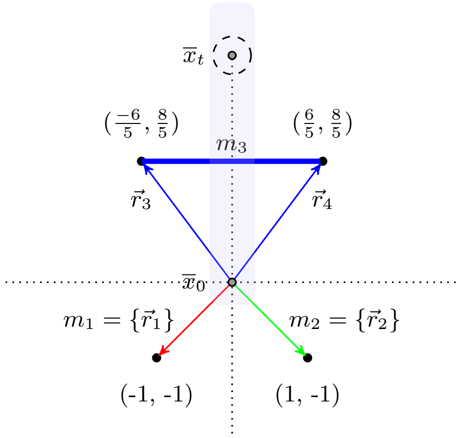


Figure 1: Bounded-rate multi-mode system with three modes and two variables.

for bounded-rate multi-mode systems. We show that the problem is co-NP complete. Moreover, we show that it is fixed parameter tractable, i.e., if the number of dimensions is fixed, then the robust reachability problem can be solved in polynomial time.

Our existence proofs are constructive: in case of a positive answer, we can also give a dynamic schedule that, given a tolerance level $\varepsilon{>}0$, guarantees reachability of an open ball of ε radius around the target state in finitely many steps. It is then simple to extend these results to different path planning problems. We discuss the extension of the robust reachability problem to motion planning, and exploit our results to provide an alternative and simpler proof for the decidability of the robust schedulability problem. We also show that this problem can be solved in polynomial time for systems with fixed dimension, improving the result [4] where authors only give a polynomial algorithm to decide 2-dimensional systems. We notice that these results can be combined to *stable reachability*, where the goal is to first reach an ε ball around a target, and then stay in this ball for ever.

EXAMPLE 1. *An example of a bounded-rate multi-mode system with two variables, say x and y, and three modes m_1, m_2, and m_3, is given in the Figure 1. Modes m_1 and m_2 are precise, while mode m_3 is uncertain, and environment can give any rate vector that is a convex combination of rate vectors $\vec{r}_3$ and $\vec{r}_4$. The safety set is given as the blue rectangle. The reachability problem here is to decide whether, for every $\varepsilon > 0$, controller has a sequence of time delays and choice of modes such that no matter what rate is given by the environment the system reaches a state in ε-neighborhood of $\overline{x}_t$. The schedulability problem asks whether the scheduler has an infinite non-Zeno sequence of choices of modes and time delays such that the system always stays within the safety set, while stable reachability problem asks for a strategy to first reach an ε-neighborhood of $\overline{x}_t$ and then to stay in that neighborhood using a non-Zeno strategy.*

We also consider the reachability problem with minimum dwell-time requirement and show that in the absence of the safety set the problem is undecidable for arbitrary bounded-rate multi-mode systems, but turns out to be decidable for systems with non-negative rates. We also study the problem of the existence of discrete control where controller is required to choose modes at times multiple of a given sampling rate. We show that the reachability problem is EXPTIME-complete for this class of controllers. Finally, we show that adding very simple structure to bounded-rate multi-mode systems by introducing clock variables (variables with precise uniform rates in each mode)—that appear as guards on the transitions and can be reset on the discrete transitions—leads to undecidability of the robust reachability problem.

Our algorithm can be combined with algorithms to explore non-convex high-dimensional spaces, such as rapidly exploring random tree (RRT) algorithm [11], to yield robust control for such systems. Intuitively, RRT algorithm can return a path from the source to the destination by random exploration of the state space, which can be robustly followed by repeated applications of our algorithm in context of systems modeled as bounded-rate multi-mode systems.

For a review of related work on constant-rate multi-mode systems we refer the reader to [4, 5]. Le Ny and Pappas [12] initiated work on the sequential composition of robust controller specifications. In this light, our results can be understood as an effort to analyze complexity of this problem for the system of relatively simple dynamics. There is a huge body of work on path-following and trajectory tracking of autonomous robots under uncertainty. For a detailed survey we refer the reader to [1]. There is a vast literature on decidable subclasses of hybrid automata [2, 7]. Most notable among these classes are initialized rectangular hybrid automata [9], two-dimensional piecewise-constant derivative systems [6], and timed automata [3].

The paper is organized as follows. We begin by formal definition of the problem in the next section, followed by the proof of our key result in Section 3. We present some applications of our main algorithm to solve schedulability, stable reachability, and path following problems in Section 4. In Section 5 we present results regarding bounded-rate multi-mode systems with discrete controller and dwell-time requirements. We conclude the paper by discussing results on generalized model in Section 6.

2. ROBUST REACHABILITY PROBLEM

Prior to formally introducing the robust reachability problem for multi-mode systems, we set the notation used in the rest of the paper and recall some standard results.

2.1 Preliminaries

Points and Vectors. Let $\mathbb{R}$ be the set of real numbers. We represent the states in our system as points in $\mathbb{R}^n$ that is equipped with the standard *Euclidean norm* $\|\cdot\|$. We denote points in this state space by $\overline{x}, \overline{y}$, vectors by $\vec{r}, \vec{v}$, and the i-th coordinate of point $\overline{x}$ and vector $\vec{r}$ by $\overline{x}(i)$ and $\vec{r}(i)$, respectively. We write $\vec{0}$ for a vector with all its coordinates equal to 0; its dimension is often clear from the context. The distance $\|\overline{x}, \overline{y}\|$ between points $\overline{x}$ and $\overline{y}$ is defined as $\|\overline{x} - \overline{y}\|$.

Boundedness and Interior. We denote an *open ball* of radius $d \in \mathbb{R}_{\geq 0}$ centered at $\overline{x}$ as $B_d(\overline{x}) = \{\overline{y} \in \mathbb{R}^n \ : \ \|\overline{x}, \overline{y}\| < d\}$. We denote a closed ball of radius $d \in \mathbb{R}_{\geq 0}$ centered at $\overline{x}$ as $\overline{B_d(\overline{x})}$. We say that a set $S \subseteq \mathbb{R}^n$ is *bounded* if there exists $d \in \mathbb{R}_{\geq 0}$ such that, for all $\overline{x}, \overline{y} \in S$, we have $\|\overline{x}, \overline{y}\| \leq d$. The *interior* of a set S, $\text{int}(S)$, is the set of all points $\overline{x} \in S$, for which there exists $d > 0$ s.t. $B_d(\overline{x}) \subseteq S$.

Convexity. A point $\overline{x}$ is a *convex combination* of a finite set of points $X = \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_k\}$ if there are $\lambda_1, \lambda_2, \ldots, \lambda_k \in [0,1]$ such that $\sum_{i=1}^{k} \lambda_i = 1$ and $\overline{x} = \sum_{i=1}^{k} \lambda_i \cdot \overline{x}_i$. The *convex hull* of X is the set of all points that are convex combinations of points in X. We say that $S \subseteq \mathbb{R}^n$ is *convex* iff, for all $\overline{x}, \overline{y} \in S$ and all $\lambda \in [0,1]$, we have $\lambda\overline{x} + (1-\lambda)\overline{y} \in S$ and moreover, S is a *convex polytope* if it is bounded and there exists $k \in \mathbb{N}$, a matrix A of size $k \times n$ and a vector $\vec{b} \in \mathbb{R}^k$ such that $\overline{x} \in S$ iff $A\overline{x} \leq \vec{b}$.

A point $\overline{x}$ is a *vertex* of a convex polytope P if it is not a convex combination of two distinct (other than $\overline{x}$) points in P. For a convex polytope P we write $\mathsf{vert}(P)$ for the finite set of points that correspond to the vertices of P. Each point in P can be written as a convex combination of the points in $\mathsf{vert}(P)$. In other words, P is the *convex hull* of $\mathsf{vert}(P)$.

2.2 Multi-Mode Systems

A multi-mode system is a hybrid system, or rather a switched system, equipped with finitely many *modes* and finitely many real-valued *variables*. A configuration is described by the values of the variables. These values change as time elapses at the rates determined by the modes being used. The choice of the rates is nondeterministic, which introduces a notion of adversarial behavior.

Definition 1 (Multi-Mode Systems). *A multi-mode system is a tuple $\mathcal{H} = (M, n, \mathcal{R})$ where: M is the finite nonempty set of* modes, *n is the number of continuous variables, and $\mathcal{R} : M \to 2^{\mathbb{R}^n}$ is the rate-set function that, for each mode $m \in M$, gives a set of vectors. We often write $\vec{r} \in m$ for $\vec{r} \in \mathcal{R}(m)$ when $\mathcal{R}$ is clear from the context.*

A finite *run* of a multi-mode system $\mathcal{H}$ is a finite sequence of states, timed moves, and rate vector choices $\varrho = \langle \overline{x}_0, (m_1, t_1), \vec{r}_1, \overline{x}_1, \ldots, (m_k, t_k), \vec{r}_k, \overline{x}_k \rangle$ s.t., for all $1 \leq i \leq k$, we have $\vec{r}_i \in \mathcal{R}(m_i)$ and $\overline{x}_i = \overline{x}_{i-1} + t_i \cdot \vec{r}_i$. For such a run ϱ we say that $\overline{x}_0$ is the *starting state*, while $\overline{x}_k$ is its *last state*. An infinite run is defined in a similar manner. We write *Runs* and *FRuns* for the set of infinite and finite runs of $\mathcal{H}$, and *Runs*$(\overline{x})$ and *FRuns*$(\overline{x})$ for the set of infinite and finite runs of $\mathcal{H}$ that start from $\overline{x}$.

An infinite run $\langle \overline{x}_0, (m_1, t_1), \vec{r}_1, \overline{x}_1, (m_2, t_2), \vec{r}_2, \ldots \rangle$ is *Zeno* if $\sum_{i=1}^{\infty} t_i < \infty$. Given a set $S \subseteq \mathbb{R}^n$ of safe states, we say that a run $\langle \overline{x}_0, (m_1, t_1), \vec{r}_1, \overline{x}_1, (m_2, t_2), \vec{r}_2, \ldots, (m_k, t_k), \vec{r}_k, \overline{x}_k \rangle$ is S-safe if $\overline{x}_i \in S$ for all $0 \leq i \leq k$; and for all $0 \leq i < k$ we have that $\overline{x}_i + t \cdot \vec{r}_{i+1} \in S$ for all $t \in [0, t_{i+1}]$, assuming $t_0 = 0$. Notice that, if S is a convex set and $\overline{x}_i \in S$ for all $i \geq 0$, then this holds iff $\overline{x}_i \in S$ for all $0 \leq i \leq k$. Sometimes we simply call a run safe when the safety set is clear from the context.

We formally give the semantics of a multi-mode system $\mathcal{H}$ as a turn-based two-player game between two players, *scheduler* and *environment*, who choose their moves to construct a run of the system. The system starts in a given starting state $\overline{x}_0 \in \mathbb{R}^n$. At each turn, the scheduler chooses a timed move, a pair $(m, t) \in M \times \mathbb{R}_{>0}$ consisting of a mode and a time duration, and the environment chooses a rate vector $\vec{r} \in m$ and as a result the system changes its state from $\overline{x}_0$ to the state $\overline{x}_1 = \overline{x}_0 + t \cdot \vec{r}$ in t time units following the linear trajectory according to the rate vector $\vec{r}$. From the next state, $\overline{x}_1$, the scheduler again chooses a timed move and the environment an allowable rate vector, and the game continues forever in this fashion. The focus of this paper is on robust *reachability* problem where, given a starting state $\overline{x}_0$, a target vertex $\overline{x}_t$, a bounded and convex safety set S and tolerance $\varepsilon > 0$, the goal of the scheduler is to visit a state in an open ball of radius ε centered at $\overline{x}_t$ via an S-safe run. The goal of the environment is the opposite.

Given a bounded and convex safety set S and tolerance $\varepsilon > 0$, we define the *robust reachability objective* $\mathcal{W}^S_{\text{Reach}}(\overline{x}_t, \varepsilon)$ as the set of infinite runs of $\mathcal{H}$ that visit a state in $B_\varepsilon(\overline{x}_t)$. In a reachability game the winning objective of the scheduler is to make sure that the constructed run of a system belongs to $\mathcal{W}^S_{\text{Reach}}(\overline{x}_t, \varepsilon)$, while the goal of the environment is the opposite. The choice selection mechanism of the players is typically defined as strategies. A *strategy* σ of the scheduler is function $\sigma{:}\mathit{FRuns} {\to} M {\times} \mathbb{R}_{\geq 0}$ that gives a timed move for every history of the game. A strategy π of the environment is a function $\pi : \mathit{FRuns} \times (M \times \mathbb{R}_{\geq 0}) \to \mathbb{R}^n$ that chooses an allowable rate for a given history of the game and choice of the scheduler. We write Σ and Π for the set of strategies of the scheduler and the environment, respectively.

Given a starting state $\overline{x}_0$ and a strategy pair $(\sigma, \pi) \in \Sigma \times \Pi$ we define the unique run $\mathit{Run}(\overline{x}_0, \sigma, \pi)$ starting from $\overline{x}_0$ as

$$\mathit{Run}(\overline{x}_0, \sigma, \pi) = \langle \overline{x}_0, (m_1, t_1), \vec{r}_1, \overline{x}_1, (m_2, t_2), \vec{r}_2, \ldots \rangle$$

where, for all $i \geq 1$, $(m_i, t_i) = \sigma(\langle \overline{x}_0, (m_1, t_1), \vec{r}_1, \overline{x}_1, \ldots, \overline{x}_{i-1} \rangle)$ and $\vec{r}_i = \pi(\langle \overline{x}_0, (m_1, t_1), \vec{r}_1, \overline{x}_1, \ldots, \overline{x}_{i-1}, m_i, t_i \rangle)$ and $\overline{x}_i = \overline{x}_{i-1} + t_i \cdot \vec{r}_i$. The scheduler wins the game if there is a $\sigma \in \Sigma$ such that, for all $\pi \in \Pi$, we get $\mathit{Run}(\overline{x}_0, \sigma, \pi) \in \mathcal{W}^S_{\text{Reach}}(\overline{x}_t, \varepsilon)$. Such a strategy σ is *winning*. Similarly, the environment wins the game if there is $\pi \in \Pi$ such that for all $\sigma \in \Sigma$ we have $\mathit{Run}(\overline{x}_0, \sigma, \pi) \notin \mathcal{W}^S_{\text{Reach}}(\overline{x}_t, \varepsilon)$. Again, π is called *winning* in this case. If a winning strategy for scheduler exists, we say that the state $\overline{x}_t$ is ε-reachable from the state $\overline{x}_0$ for given safety set S and tolerance ε. We also say that the state $\overline{x}_t$ is *robustly reachable* from $\overline{x}_0$ if it is ε-reachable for all $\varepsilon > 0$. The following is the main algorithmic problem studied in this paper.

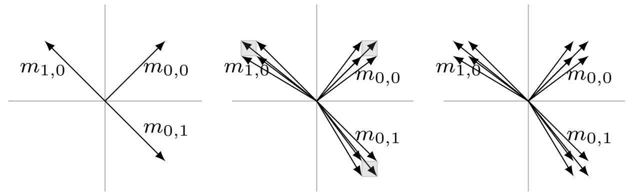
Definition 2 (Robust Reachability). *Given a multi-mode system $\mathcal{H}$, a convex safety set S, a starting state $\overline{x}_0 \in \mathrm{int}(S)$, and a target state $\overline{x}_t \in \mathrm{int}(S)$, decide whether $\overline{x}_t$ is robustly reachable from $\overline{x}_0$.*

To algorithmically decide the robust reachability problem, we need to restrict the range of $\mathcal{R}$ and the domain of the safety set S in a robust reachability game on a multi-mode system. The most general model that we consider is the bounded-rate multi-mode systems (BMS).

Definition 3 (Bounded-Rate Systems). *A bounded-rate multi-mode system (BMS)is multi-mode system $\mathcal{H} = (M, n, \mathcal{R})$ such that $\mathcal{R}(m)$ is a convex polytope for every $m \in M$. We also assume that the safety set S is specified as a convex polytope.*

For every mode $m_i \in M$ of a BMS we assume an arbitrary but fixed ordering on the vertices of $\mathcal{R}(m)$. By exploiting the notations slightly, it allows us to write $\mathcal{R}(m_i)(j)$ for the rate vector corresponding to j-th vertex of mode m_i. When there is no confusion, we also write $\mathcal{R}(i)(j)$ for $\mathcal{R}(m_i)(j)$.

In our proofs we often refer to another variant of multi-mode systems, in which there are only a fixed number of different rates in each mode (i.e., $\mathcal{R}(m)$ is finite for all $m \in M$). We call such a multi-mode system *multi-rate multi-mode systems* (MMS). Finally, a special form of MMS are *constant-rate multi-mode systems* (CMS) [5], in which $\mathcal{R}(m)$ is a singleton for all $m \in M$. We sometimes use $\mathcal{R}(m)$



(a) Constant-Rate (b) Bounded-Rate (c) Multi-Rate

Figure 2: Restricted Multi-mode Systems

to refer to the unique element of the set $\mathcal{R}(m)$ in a CMS. The concepts related to the robust reachability games for BMS and MMS are already defined for multi-mode systems. Similar concepts also hold for CMS but with no real choice for the environment. Examples of CMS, BMS, and MMS are shown in Figure 2.

We say that a CMS $H = (M, n, R)$ is an instance of a multi-mode system $\mathcal{H} = (M, n, \mathcal{R})$ if for every $m \in M$ we have that $R(m) \in \mathcal{R}(m)$. For example, the CMS shown in Figure 2.(a) is an instance of BMS in Figure 2.(b). We denote the set of instances of a multi-mode system $\mathcal{H}$ by $[\![\mathcal{H}]\!]$. Notice that for a BMS $\mathcal{H}$, the set $[\![\mathcal{H}]\!]$ of its instances is uncountable (unless the BMS is a CMS), while for an MMS $\mathcal{H}$ the set $[\![\mathcal{H}]\!]$ is finite, and exponential in the size of $\mathcal{H}$. We say that an MMS $(M, n, \mathcal{R}')$ is the *extreme-rate* MMS of a BMS $(M, n, \mathcal{R})$ if $\mathcal{R}'(m) = \mathsf{vert}(\mathcal{R}(m))$. The MMS in Figure 2.(c) is the extreme-rate MMS for the BMS in Figure 2.(b) We write $\mathit{Ext}(\mathcal{H})$ for the extreme-rate MMS of the BMS $\mathcal{H}$.

The following theorem is the key observation of the paper.

THEOREM 1. *Given a BMS $\mathcal{H} = (M, n, \mathcal{R})$, convex safety set S, starting state $\overline{x}_0 \in \mathrm{int}(S)$ and target state $\overline{x}_t \in \mathrm{int}(S)$, the target state $\overline{x}_t$ is robustly reachable if and only if for every CMS in $[\![\mathit{Ext}(\mathcal{H})]\!]$ the state $\overline{x}_t$ is reachable from $\overline{x}_0$.*

Alur et al. [5] presented a polynomial-time algorithm to decide if a state $\overline{x}_t$ is reachable from a starting state $\overline{x}_0$ for CMS. In particular, for starting and target states in the interior of the safety set, they characterized a necessary and sufficient condition.

THEOREM 2 ([5]). *The scheduler has a winning strategy in a CMS (M, n, R), with convex safety set S and starting state $\overline{x}_0 \in \mathrm{int}(S)$ and target state $\overline{x}_t \in \mathrm{int}(S)$, if and only if there is $\vec{t} \in \mathbb{R}_{\geq 0}^{|M|}$ satisfying:*

$$\overline{x}_0 + \sum_{i=1}^{|M|} R(i)(j) \cdot \vec{t}(i) = \overline{x}_t \text{ for } 1 \leq j \leq n. \tag{1}$$

Notice that in such a case controller has a strategy to reach the target state precisely. The intuition behind Theorem 2 is that the scheduler has a winning strategy if and only if it is possible to reach the target state from the starting state in using a combination of the rate vectors.

Using Theorems 1 and 2 it follows that the robust reachability problem is in co-NP. By reducing the validity checking problem of propositional logic formulas in DNF, we show that the robust reachability problem for BMS is indeed complete the class co-NP. On a positive side, we show that the robust reachability problem for BMS and CMS is fixed parameter tractable, i.e. it is polynomial for fixed number of variables. It brings us to our next key result.

THEOREM 3 (COMPLEXITY). *The robust reachability problems for BMS and CMS are co-NP complete. However, it is fixed parameter tractable with fixed number of variables.*

3. DECIDABILITY AND COMPLEXITY

This section is dedicated to the proofs of Theorem 1 and Theorem 3.

3.1 Proof of Theorem 1

We prove Theorem 1 by showing that the condition is necessary and sufficient in the following two lemmas.

LEMMA 4. *Given a BMS $\mathcal{H}$, safety set S, starting state $\overline{x}_0 \in \mathrm{int}(S)$ and target state $\overline{x}_t \in \mathrm{int}(S)$, the target state is not robustly reachable if there exists a CMS (M, n, R) in $[\![\mathit{Ext}(\mathcal{H})]\!]$ for which $\overline{x}_t$ is not reachable from $\overline{x}_0$.*

PROOF. From Theorem 2, we have that an interior point $\overline{x}_t$ of S is reachable iff it is in the conical hull of rates in the CMS. Let $\mathcal{K}$ denote this connical hull. Note that $\mathcal{K}$ is closed and that, by our assumption, $\overline{x}_t \notin \mathcal{K}$. This implies that the distance $\varepsilon = \inf_{\overline{x} \in \mathcal{K}} \|\overline{x}_t, \overline{x}\|$ between $\overline{x}_t$ and $\mathcal{K}$ is positive. Consequently, $B_\epsilon(\overline{x}_t)$ and $\mathcal{K}$ are disjoint. It follows that when the environment follows the strategy to choose the rate $R(m)$ when presented with a mode m, then the controller cannot reach an ϵ ball around $\overline{x}_t$. $\square$

LEMMA 5. *Given a BMS $\mathcal{H}$, safety set S, starting state $\overline{x}_0 \in \mathrm{int}(S)$ and target state $\overline{x}_t \in \mathrm{int}(S)$, the target state is robustly reachable if for all CMS (M, n, R) in $[\![\mathit{Ext}(\mathcal{H})]\!]$ the state $\overline{x}_t$ is reachable from $\overline{x}_0$.*

We give a constructive proof of this lemma by constructing an algorithm (Algorithm 1) giving a strategy of the player to reach $B_\epsilon(\overline{x}_t)$ for a given $\epsilon > 0$.

Before we elaborate on the working of the algorithm, we need to explain the idea of projections. In our algorithms we sometimes represent a point $\overline{x} \in \mathbb{R}^n$ by explicitly defining its projection towards the direction vector $\vec{v} = \overline{x}_t - \overline{x}_0$ and (small) projections towards extreme rate vectors of various modes.

DEFINITION 4 (PROJECTION). *Given a BMS $(M, n, \mathcal{R})$ we say that a tuple (λ, π) is a* projection *of a point $\overline{x}$, where $\lambda \in \mathbb{R}_{\geq 0}$ is the projection towards $\vec{v}$ and $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_{\geq 0}$ is the projection towards extreme rate-vectors of various modes, such that $\pi(i, j)$ is the projection towards j-th vertex of the rate polytope $\mathcal{R}(m_i)$, if:*

$$\overline{x} = \lambda \cdot \vec{v} + \sum_{i=1}^{|M|} \sum_{j=1}^{|\mathsf{vert}(\mathcal{R}(m_i))|} \pi(i, j) \cdot \mathcal{R}(i, j).$$

Notice that such projections are often not unique. We write the (λ_0, π_0) for the projection such that $\lambda_0 = 0$ and $\pi_0(i, j) = 0$ for all i, j. Given a projection $P = (\lambda, \pi)$ of a state $\overline{x}$ we say that $\pi(i, j)$ is the contribution of the j^{th} vertex of the rate polytope of mode m_i. We also say that a vertex $\mathcal{R}(i, j)$ does not contribute in a projection P if $\pi(i, j) = 0$, while we say that a mode does not contribute in a projection P if $\pi(i, j) = 0$ holds for all corners j of $\mathcal{R}(m_i)$.

Given a tolerance level of ϵ, the strategy for the player to reach $B_\epsilon(\overline{x}_t)$ is given by Algorithm 1. A feature of the algorithm is that the player selects time τ at every step. It calls function NEXTMODE described in Algorithm 2 to get the mode that the player chooses. Depending on the choice of the environment, the current point is updated. This process goes on until an ϵ ball around $\overline{x}_t$ is reached.

Algorithm 1: Dynamic reachability algorithm

Input: BMMS $\mathcal{H}$, starting state $\overline{x}_0$, tolerance level ϵ
Output: Reachability Algorithm

$\mathsf{B} := \max_{m\in M} \max_{\vec{r}\in\mathcal{R}(m)} \|\vec{r}\|$;
$\overline{x} := \overline{x}_0$, the current point;
$\vec{v} := \overline{x}_t - \overline{x}_0$, the reachability direction;
$\gamma_1 :=$ the shortest distance of $\overline{x}_0$ from borders of S;
$\gamma_2 :=$ the shortest distance of $\overline{x}_t$ from borders of S;
$\tau := \min(\epsilon/2, \gamma_1, \gamma_2)/(\mathsf{B}|M|)$;
$\sigma :=$ an array, one element for each CMS $\mathcal{F}$ in $[\![Ext(\mathcal{H})]\!]$ s.t $\sigma(\mathcal{F}) = \vec{t}$ s.t. $\vec{t}$ is a solution for the CMS $\mathcal{F}$ for reachability to $\vec{v}$;
Projection $P = (\lambda_0, \pi_0)$;
while $\|\overline{x} - \overline{x}_t\| > \epsilon$ **do**
  P, $m = \text{NEXTMODE}(P, \vec{v}, \sigma)$;
  $\vec{r} = \text{SENSECURRENTRATE}(\overline{x}, m, \tau)$;
  $\overline{x} = \overline{x} + \tau\vec{r}$;
  $P = \text{UPDATEPROJECTION}(P, m, \vec{r})$;

Algorithm 2: NEXTMODE$(P, \vec{v}, \sigma)$

Input: Projection P, reachability direction $\vec{v}$, reachability solution array σ
Output: Projection P, mode m

while *true* **do**
  if *there exists mode m_i having zero contribution to Projection P* **then**
    return P, m_i;
  else
    $R(m_i) := \mathsf{vert}(\mathcal{R}(m_i))(j)$ such that $\mathsf{vert}(\mathcal{R}(m_i))(j)$ has non-zero contribution to P for each i;
    CMS $\mathcal{F} := (M, n, R)$ is the corresponding instance of $\mathcal{H}$;
    $P = \text{REDUCECOMP}(\mathcal{F}, \sigma(\mathcal{F}), P)$;

The job of NEXTMODE function is to nullify the contribution of a mode m by expressing the point in a different way. It calls upon REDUCECOMP function described in Algorithm 3 to achieve this. The correctness of the Algorithm 3 follows from the following proposition.

PROPOSITION 6. *Every non-negative linear combination of rates of a CMS $\mathcal{F} = (M, n, R) \in [\![Ext(\mathcal{H})]\!]$ that is reachable to $\vec{v}$ can be written as the sum of a non-negative component along $\vec{v}$ and a non-negative linear combination of the rates where contribution of one of the rates is* 0.

PROOF. Given a CMS $\mathcal{F} = (M, n, R)$ as an instance of $[\![Ext(\mathcal{H})]\!]$, we have

$$\vec{v} = \sum_{i=1}^{|M|} \sigma(\mathcal{F})(i).R(m_i) \tag{2}$$

Given any non-negative linear combination of vectors in R, $\sum_{i=1}^{|M|} c_i R(m_i)$, $c_i \geq 0$, let $k = \arg\min_{i,\sigma(\mathcal{F})(i)>0}(c_i/\sigma(\mathcal{F})(i))$.

Algorithm 3: REDUCECOMP$(\mathcal{F}, \sigma(\mathcal{F}), P)$

Input: CMS $\mathcal{F} = (M, n, R)$, solution time vector for the CMS $\sigma(\mathcal{F})$, current Projection $P = (\lambda, \pi)$
Output: Projection $P' = (\lambda', \pi')$ s.t. contribution of one of the rates in $\mathcal{F}$ has been nullified

$(\lambda', \pi') = (\lambda, \pi)$;
$k := \arg\min_{i,\sigma(\mathcal{F})(i)>0} (\pi(i, R(i))/\sigma(\mathcal{F})(i))$;
$\lambda' = \lambda + \pi(k, R(m_k))/\sigma(\mathcal{F})(k)$;
for i *in* $\{1, 2, \ldots, |M|\}$ **do**
  $\pi'(i, R(i)) = \pi(i, R(i)) - (\pi(k, R(k)) * \sigma(\mathcal{F})(i))/\sigma(\mathcal{F})(k)$;
return (λ', π');

Algorithm 4: UPDATEPROJECTION$(P, m, \vec{r})$

Input: current Projection $P = (\lambda, \pi)$, mode m_i, rate $\vec{r}$
Output: Projection $P = (\lambda', \pi')$ with rate r taken for time τ

$(\lambda', \pi') = (\lambda, \pi)$;
$\vec{r} = \sum_{j=1}^{|vert(\mathcal{R}(m_i))|} \theta_j vert(\mathcal{R}(m_i))(j)$, the convex combination of vertex of the rate polytope of mode m_i;
for j *in* $\{1, 2, \ldots, |vert(\mathcal{R}(m_i))|\}$ **do**
  $\pi'(i, j) = \theta_j * \tau$;
return (λ', π');

The following calculations show that every non-negative linear combination of rates of a CMS $\mathcal{F} = (M, n, R)$ that is reachable to $\vec{v}$ can be written as the sum of a non-negative component along $\vec{v}$ and a non-negative linear combination of the rates where contribution of one of the rates is 0. For clarity, $\sigma(\mathcal{F})(i)$ has been written as σ_i below.

$$\begin{aligned}
\sum_{i=1}^{|M|} c_i R(m_i) &= \sum_{i=1}^{|M|}(c_i - \frac{c_k\sigma_i}{\sigma_k} + \frac{c_k\sigma_i}{\sigma_k})R(m_i) \\
&= \sum_{i=1}^{|M|}(c_i - \frac{c_k\sigma_i}{\sigma_k})R(m_i) + \sum_{i=1}^{|M|}\frac{c_k\sigma_i}{\sigma_k}R(m_i) \\
&= \sum_{i=1}^{|M|}(c_i - \frac{c_k\sigma_i}{\sigma_k})R(m_i) + \frac{c_k}{\sigma_k}\sum_{i=1}^{|M|}\sigma_i R(m_i) \\
&= \sum_{i=1}^{|M|}(c_i - \frac{c_k\sigma_i}{\sigma_k})R(m_i) + \frac{c_k}{\sigma_k}\vec{v}
\end{aligned}$$

The last step follows from Equation 2. Note that, since $k = \arg\min_{i,\sigma_i>0}(c_i/\sigma_i)$, we have that $c_i - \frac{c_k\sigma_i}{\sigma_k} \geq 0$ holds for each i and $= 0$ holds for $i = k$. $\square$

This explains the working of Algorithm 3. We say that the contribution of $R(m_k)$ is consumed in this process. Every invocation of Algorithm 3 consumes at least one corner of one of the modes in M. Hence, it guarantees that after some finite iterations, some mode will be consumed in the process. This proves the termination of Algorithm 2.

PROPOSITION 7 (SAFETY). *All the states visited during an execution of Algorithm 1 are strictly inside the safety set.*

PROOF. We will demonstrate that all the point visits during a run belong to the safety set. We first claim that the point reached at any step in the algorithm, x, can be written as the sum of a non-negative component along $\vec{v}$ and small components along some rate in each of the modes. Formally,

$$\overline{x} = \lambda\vec{v} + \sum_{i=1}^{|M|} t_i \vec{r}_i, \tag{3}$$

where $\lambda \geq 0$, $\vec{r}_i \in \mathcal{R}(m_i)$, $0 \leq t_i \leq \tau$ for all modes $m_i \in M$. We prove this by induction on the number of steps. The initial point $x_0 = 0$ is trivially written in the above form with $\lambda = 0$, $t_i = 0$ and r_i, any rate vector in mode m_i for all i. If, after j steps, $\overline{x} = \lambda\vec{v} + \sum_{i=1}^{|M|} t_i\vec{r}_i$ with $\lambda \geq 0$, $t_i \geq 0$ for all i, then Algorithm 2 ensures that $\overline{x}$ can be written in an alternative way such that the contribution of some mode m_k is 0 in $\overline{x}$. In this process, λ is non-decreasing and the contribution of other modes is non-increasing but always ≥ 0. This provides $\overline{x} = \lambda_1\vec{v} + \sum_{i\in[|M|]\setminus\{k\}} t'_i\vec{r}'_i$ with $\lambda_1 \geq \lambda, 0 \leq t'_i \leq t_i$. The mode chosen by the player in this step is m_k and the time chosen is τ, the new point reached is $\overline{x}' = \overline{x} + \sum_{i=1}^{|M|} t'_i\vec{r}'_i$, where $t'_k = \tau$ and $\vec{r}'_k$ is the rate chosen by the environment in mode m_k. So, we again have $\overline{x}'$ in the form specified by Equation 3.

$$\begin{aligned} |\overline{x} - \lambda\vec{v}| &= |\sum_{i=1}^{|M|} t_i r_i| \leq \sum_{i=1}^{|M|} |t_i r_i| \\ &\leq \sum_{i=1}^{|M|} \tau\mathsf{B} \leq \sum_{i=1}^{|M|} \frac{\min(\epsilon/2, \gamma_1, \gamma_2)}{|M|} = \min(\epsilon/2, \gamma_1, \gamma_2). \end{aligned}$$

We have used the value of τ defined in Algorithm 1. The last equation also shows that the current point is inside a ball of radius $\epsilon/2$ from the point $\lambda\vec{v}$.

We now prove that $\lambda \leq 1 + \epsilon/(2\|\vec{v}\|)$. We prove this by induction on the number of steps. Initially, $\lambda = 0 \leq 1 + \epsilon/(2\|\vec{v}\|)$. Let $\overline{x}_j$ and $\overline{x}_{j+1}$ be the point reached after j and $j+1$ steps, respectively. After j steps, if $\lambda_j \leq 1 + \epsilon/(2\|\vec{v}\|)$ and we are not inside an ϵ ball around $\overline{x}_t$, then by geometry, $\lambda_j \leq 1 - \epsilon/(2\|\vec{v}\|)$. Let λ_j and λ_{j+1} be the respective projection along direction $\vec{v}$. Suppose, some rate $\vec{r}$ is taken for time τ. Then,

$$\|\overline{x}_{j+1} - \overline{x}_j\| = \|\tau r\| \leq \tau\|r\| \leq \tau\mathsf{B} \leq \frac{\epsilon}{2|M|}$$

So, successive points differ by a distance of at most $\epsilon/2|M| \leq \epsilon/2$ and they are centered around $\lambda_j\vec{v}$ and $\lambda_{j+1}\vec{v}$ in a ball of radius $\epsilon/2$. Since $\|\lambda_{j+1}\vec{v} - \lambda_j\vec{v}\| \leq \epsilon$ we have that

$$\lambda_{j+1} \leq \lambda_j + \frac{\epsilon}{\|\vec{v}\|} \leq 1 + \frac{\epsilon}{(2\|\vec{v}\|)}$$

The last step follows from $\lambda_j \leq 1 - \epsilon/(2\|\vec{v}\|)$ as argued already. So, we have proved by induction that $\lambda \leq 1 + \epsilon/(2\|\vec{v}\|)$. Therefore, at any step in the algorithm, $\lambda\vec{v}$ is a convex combination of $\overline{x}_0$ and $\overline{x}_t + \epsilon\vec{v}/(2\|\vec{v}\|)$. Therefore, a ball of radius $\min(\epsilon/2, \gamma_1, \gamma_2)$ lies completely inside the safety set S. This follows from the definition of γ_1 and γ_2. So, the algorithm is safe. □

Proposition 8 (Termination). *The Algorithm 1 always terminates.*

Proof. We show that the algorithm terminates in finitely many steps by demonstrating the progress towards the target state. We say that τ of mode m is pumped in Projection P at step j of the algorithm, if the player chooses mode m at step j. We show that there exists $\delta > 0$ such that, in every $|M| + 1$ steps of the algorithm, the variable λ of the current point P increases by at least δ. Consider $|M| + 1$ consecutive runs of the algorithm. Since we pump a mode at every step of the algorithm, there is a mode m_i, which was chosen twice for pumping. This implies that at least τ of m_i was consumed in n steps. So, at least $\tau/|\mathsf{vert}(\mathcal{R}(m_i))|$ was consumed by some vertex $\mathcal{R}(m_i)(j)$ of $\mathcal{R}(m_i)$. Every CMS with $R(m_i) = \mathcal{R}(m_i)(j)$ guarantees a fixed increase δ_1 in λ for $\tau/|vert(\mathcal{R}(m_i))|$ consumed of corner $\mathcal{R}(m_i)(j)$, since the solution vector $\sigma(\mathcal{F})$ for every CMS $\mathcal{F}$ is fixed. Hence, δ equals the mininum of δ_1 over all vertices of all modes is the least increase in λ. This is the minimum increase in $n+1$ steps of the algorithm. Hence, progress is proved.

Now, we show termination. Progress of δ in every $|M|+1$ iterations along with the condition $\lambda \leq 1 + \epsilon/(2\|\vec{v}\|)$ and increase in λ bounded by ϵ in every step guarantee that, after some finite iterations, $1-\epsilon/(2\|\vec{v}\|) \leq \lambda \leq 1+\epsilon/(2\|\vec{v}\|)$, in which case $\|\overline{x} - \lambda\vec{v}\| \leq \epsilon/2$ implies $\overline{x} \in B_\epsilon(\overline{x}_t)$. □

The proof of Lemma 5 is now complete.

3.2 Proof of Theorem 3

With the two lemmas in place, we can proceed with proving the main complexity results. We start with the positive result that states that the problem is tractable in practice.

Theorem 9. *The reachability problem for* BMS *and* MMS *is fixed parameter tractable, where the parameter is the number of variables. In particular, it is polynomial for* BMS *and* MMS *with fixed dimension d.*

Proof. We first observe that, for fixed dimensions, the number of extreme points per mode is polynomial in the size of the defining matrix. Thus, $Ext(\mathcal{H})$ is polynomial in $\mathcal{H}$ for BMS $\mathcal{H}$.

For the sake of simplicity assume that the starting point is the origin and we wish to reach position $\overline{p}$.

From Lemmas 4 and 5, we can infer that the existence of a $\mathsf{CMS} \in [\![Ext(\mathcal{H})]\!]$ for which $\overline{p}$ is not reachable from the origin is a necessary and sufficient criterion for refuting reachability. By Theorem 2, for a CMS $\mathcal{C}$ the state $\overline{p}$ is not reachable from the origin iff it is not in the conical hull of its vertices. This is the case, iff there is a hyperplane through the origin that does not contain $\overline{p}$, such that the half-space without $\overline{p}$ it defines contains all vertices of $\mathcal{C}$.

Let $k \leq d$ be the dimension of the hull of $\mathsf{vert}(\mathcal{C})$.

We now distinguish two cases. First, assume that $\overline{p}$ is not in the hull of $\mathsf{vert}(\mathcal{C})$. In order to validate this, we can simply take $k < d$ vectors of $\mathcal{C}$ and validate that they are a basis of $\mathsf{vert}(\mathcal{C})$.

Now we assume that $\overline{p}$ is in the hull of $\mathsf{vert}(\mathcal{C})$. We now work in $\mathsf{vert}(\mathcal{C})$. There are $k-1$ vectors in $\mathsf{vert}(\mathcal{C})$ that define a hyperplane in this hull, s.t. the half-space without $\overline{p}$ it defines contains all vertices of $\mathcal{C}$. [1]

The next observation we make is that

[1]We can start with projecting the the hyperplane we started with into the hull of $\mathsf{vert}(\mathcal{C})$, and then stepwise move the hyperplane and reduce its dimension. If the space we are left with has more than one dimension, it is clear that we can at least change it to include one vertex. We can change the hyperplane to include one vector, project everything to the subspace orthogonal to this vector, and continue. The modes selected can be use to define a suitable hyperplane.

- the $k < d$ spanning vectors used from extreme points of different modes are sufficient to establish the first case in polynomial time for the MMS, because one can cheaply check that all other modes contain a vector in the space they span, while $\overline{p}$ is not a linear combination of them, and

- the $k - 1$ spanning vectors used from extreme points of different modes are sufficient to establish the second case in polynomial time for the MMS, because one can cheaply check that $\overline{p}$ is not a linear combination of them, and all other modes contain a vector in the k dimensional space spanned by them and $\overline{p}$, and $\overline{p}$ does not enter positively in the linear combination.

Thus, it suffices to perform cheap (polynomial) tests for sets of less than d vectors. The number of these sets is polynomial for fixed d. □

THEOREM 10. *The reachability problem for BMS and MMS are co-NP complete.*

PROOF. The inclusion in co-NP is implied by Lemma 4: to refute reachability, it is enough to guess a CMS $\mathcal{C}$ from $[\![\mathcal{H}]\!]$ of an MMS or $[\![Ext(\mathcal{H})]\!]$ from a BMS $\mathcal{H}$ and to verify that the target is not reachable from $\mathcal{C}$. The verification can be performed in polynomial time from Theorem 2.

We show co-NP hardness by reducing the validity checking of propositional logic formulas in DNF, where each clause is a conjunction of three literals, which refer to different propositions. We give a full proof for BMS.

Given such a formula φ with m clauses $D_1, \ldots, D_m$ and $n \geq 3$ variables $x_1, \ldots, x_n$, we construct a BMS with less than $7m + 2n + 3$ modes and $n + 3$ variables. We name n of these variables the propositions, $x_1, \ldots, x_n$, and there are three further variables, y_1, y_2, y_3, which are intuitively manipulated in three different stages of a game. Initially, all variables are 0, and the goal is to reach a state, where $y_1 = y_2 = 1$, $y_3 = n - 3$, and $x_1 = x_2 = \cdots = x_n = 0$. The safety set for all variables is the interval [-1,1].

Given $\varphi = D_1 \vee D_2 \vee \cdots \vee D_m$, where each D_i has 3 literals, we consider subclauses of $D_1, \ldots, D_m$. Each D_i has 6 non-empty subclauses. Considering the empty clause as well, we obtain $l \leq 7m + 1$ clauses $D_1, \ldots, D_m, D_{m+1}, \ldots, D_l$. Note that we do not change φ, we only need the new clauses for technical reasons. Let $N(D_i) = \{j \mid x_j \text{ or } \neg x_j \text{ occurs in } D_i\}$ for all $i = 1, \ldots, l$.

The BMS has only one nondeterministic mode, m_e, which is also the initial mode. Intuitively, the environment chooses the valuation of the variables in this mode. Our BMS allows all rate vectors with $\mathcal{R}(m_e)(x_i) \in [-1, 1]$ for $1 \leq i \leq n$, $\mathcal{R}(m_e)(y_1) = 1$, and $\mathcal{R}(m_e)(y_2) = \mathcal{R}(m_e)(y_3) = 0$. Intuitively, the environment tries to select a valuation of the variables $x_1, \ldots, x_n$ that does not satisfy φ in this mode, where the value 1 refers to 'true' and -1 refers to 'false'. m_e is the only mode with $y_1 \neq 0$. Given the goal, the controller must be in the mode m_e for exactly one time unit.

For each clause in the extended set of clauses (i.e., for $i = 1, \ldots, l$), our BMS has a clause mode, m_i. We have:

- $\mathcal{R}(m_i)(x_j) = 1$ if $\neg x_j$ occurs in D_i,

- $\mathcal{R}(m_i)(x_j) = -1$ if x_j occurs in D_i,

- $\mathcal{R}(m_i)(x_j) = 0$ for all $j \notin N(D_i)$, and

- $\mathcal{R}(m_i)(y_1) = \mathcal{R}(m_i)(y_3) = 0$, and $\mathcal{R}(m_i)(y_2) = 1$.

Intuitively, the scheduler selects a clause from $D_1, \ldots, D_m$, and resets the values of the three variable occurring in the clause to 0. The role of the additional $l - m$ clauses is to account for the capability of the environment to select values different from -1 and 1. The clause modes are the only modes with $y_2 \neq 0$. Given the goal, the scheduler must be in clause modes for exactly 1 time unit.

For each of variable x_i, our BMS has two correction modes, m_i^+ and m_i^-, and one empty correction node m_0. We have:

- $\mathcal{R}(m_i^+)(x_i) = 1$, $\mathcal{R}(m_i^+)(x_j) = 0$ for all $j \neq i$,
  $\mathcal{R}(m_i^+)(y_1) = \mathcal{R}(m_i^+)(y_2) = 0$, and $\mathcal{R}(m_i^+)(y_3) = 1$,

- $\mathcal{R}(m_0)(x_i) = 0$, for all $i = 1, \ldots, n$, $\mathcal{R}(m_0)(y_1) = \mathcal{R}(m_0)(y_2) = 0$, and $\mathcal{R}(m_0)(y_3) = 1$, and

- $\mathcal{R}(m_i^-)(x_i) = -1$, $\mathcal{R}(m_i^-)(x_j) = 0$ for all $j \neq i$,
  $\mathcal{R}(m_i^-)(y_1) = \mathcal{R}(m_i^+)(y_2) = 0$, and $\mathcal{R}(m_i^-)(y_3) = 1$.

Intuitively, the scheduler resets the values of the remaining $n - 3$ variables, not covered by the clause, to 0 using these correction modes. The correction modes are the only modes with $y_3 \neq 0$. Given the goal, the scheduler must be in correction modes for exactly $n - 3$ time units.

We first observe that the reachability problem is polynomial in φ. Next, we convince ourselves that the goal is reachable if φ is valid.

In this case, the scheduler first stays in mode m_e for one time unit. It then identifies an $i \in \{1, \ldots, m\}$ such that, for all $j \in N(D_i)$, if $x_j > 0$ then x_j is a literal of D_i and if $x_j < 0$ then $\neg x_j$ is a literal of D_i. The scheduler can then apply the clause modes for D_i and/or its subclauses for together one time unit such that, after this time unit, $x_j = 0$ holds for all $j \in N(D_i)$.

Next, the scheduler can apply, for all $j \notin N(D_i)$ the correction mode m_j^+ for x_j time units if $x_j > 0$ or m_j^- for $-x_j$ time units if $x_j < 0$. Given a clause D_i, $\big|\{j \in \{1, \ldots, n\} \mid j \notin N(D_i)\}\big| = n - 3$, this brings us to a point with $x_1 = \ldots = x_n = 0$, $y_1 = y_2 = 1$, and $y_3 \in [0, n - 3]$. From there, we can apply m_0 for $y_3 + 3 - n$ time units to reach the goal.

Finally, we have to check that, if φ is not valid, then the goal is not reachable. To see this, note that the m_e must be scheduled for exactly one time unit. The environment can therefore select a configuration that does not satisfy φ and choose rates -1 for 'false' and 1 for 'true' for this configuration each time m_0 is scheduled.

Now let us assume that the environment follows this policy, but the goal is reached. First we observe that the system must be for 1 time unit in m_e, for 1 time unit in clause modes, and for $n-3$ time units in correction modes. Clearly, some clause mode m_i is used for t time units, with $t \in [0, 1]$. Note that, if D_i refers to a clause that is satisfied by the configuration, then D_i has at most two literals. Now we observe that

- when considering the effect of the 1 time unit in m_e, we have $\sum_{j=1}^{n} |x_j| = n$,

- when considering the $1 + t$ time units the system is in m_e or a clause mode m_i, we have $\sum_{j=1}^{n} |x_j| \geq n - 2t$,

– when considering the 2 time units the system is in m_e or a clause mode m_i, we have $\sum_{j=1}^{n} |x_j| \geq n+t-3$ (no D_i exists satisfying the chosen assignment, thus $t > 0$)

– after the complete $n-1$ time units of the run, we have $\sum_{j=1}^{n} |x_j| \geq t > 0$.

This provides a contradiction to having reached the goal.

The proof can easily be extended to MMS, however we have to overcome the exponential size of the extreme-rates for m_e. In order to achieve this, we split y_1 into n variables $y_1^1, \ldots, y_1^n$ and replace m_e by n modes $m_e^1, \ldots, m_e^n$. $\mathcal{R}(m_e^i)$ has two points, where $y_1^i = 1$, $x_i \in \{-1, 1\}$ and all other $y_2 = y_3 = x_j = 0$ for all $j \neq i$. For the goal, we require $y_1^1 = \ldots = y_1^n = 1$ instead of $y_1 = 1$. The only change is that the environment now selects the values for the atomic propositions successively instead of concurrently. □

4. APPLICATIONS

In this section, we show how to apply our results for

– robust schedulability—to decide if, for all $\varepsilon > 0$, there is a non-Zeno control strategy, which guarantees that the system stays in an ε ball around the starting point;

– robust stability—to decide if, for all $\varepsilon > 0$, there is a non-Zeno control strategy, which guarantees that the system reaches an ε ball around the target point and then never leaves it again (possibly while staying in a convex safety set where the starting vertex and the target $\overline{x}_t$ are inner points); and

– robust path following—to decide if, for all $\varepsilon > 0$, a given path can be followed with ε precision.

4.1 Robust Schedulability

For ease of notation, we assume w.l.o.g. that this point is the origin $\mathbf{0}$, and we assume w.l.o.g. that $\varepsilon < 1$. The problem has been studied before in [4], but the proof we provide here is much simpler.

Robust schedulability can be derived from robust reachability by first tweaking the reachability problem slightly, such that one execution guarantees to stay within a ε-ball while consuming at least one time unit.

The central idea for adjusting a system with d variables $x_1, \ldots, x_d$ is to add one variable, c, that serves as a clock. In all rates of all modes, the rate in which this new variable progresses is 1. Next, we define the safety set as $S = \{(x_1, \ldots, x_d, c) \mid \forall i \leq d.\ 2d|x_i| \leq \varepsilon\}$, or any other convex set that does not constrain the values of c and that constraints the values of the remaining variables to be in the ε ball around $\mathbf{0}$. We now consider the problem of reaching the point $\overline{x}_t$ with $x_1 = \ldots = x_d = 0$ and $c = 1$ with ε precision. First, when projecting away the clock c, the safety set alone guarantees to be in an ε ball around $\mathbf{0}$, and second, the value of c must be greater than $1-\varepsilon$, which implies with the constant rate 1 that at least $1-\varepsilon$ time units have past.

If $\overline{x}_t$ is not robust reachable from $\mathbf{0}$, then there is an ε, for which $B_\varepsilon(\overline{x}_t)$ cannot be reached. Thus, no strategy exists to keep the system in an ε/d ball around $\mathbf{0}$ for one time unit, as this control strategy could be applied to reach the ε ball around $\overline{x}_t$. If, however, $\overline{x}_t$ is robust reachable from $\mathbf{0}$, then we can repeatedly apply such a strategy, first for ε_1, then for ε_2, and so forth, where $\varepsilon_i = 2^{-i}\varepsilon$. It is easy to see that the resulting composed strategy is non-Zeno, as all components are finite and at least one time unit passes in each component. It is also easy to see that the error can at most add up, such that one always stays in an ε ball around the starting point.

4.2 Robust Stability

Obviously, reachability to $\overline{x}_t$ and robust schedulability are prerequisites for robust stability. To see that they are also sufficient, we assuming w.l.o.g. that the ball $B_\varepsilon(\overline{x}_t)$ is contained in the safety set S. It then suffices to reach an $\overline{x}_t$ with precision $\varepsilon/2$, and then to follow a robust reachability strategy to stay in an $\varepsilon/2$ ball around the point reached.

4.3 Robust Path Following

To robustly follow a piecewise linear path with precision ε, we can simply follow the first piece with precision ε_1, the second with ε_2, and so forth, where $\varepsilon_i = 2^{-i}\varepsilon$. Following a piecewise linear path is therefore possible with arbitrary precision if each segment can be followed individually with arbitrary precision. Conversely, if one of these segments cannot be followed with arbitrary precision, then, obviously, the complete path cannot be followed with arbitrary precision. Note that the necessary and sufficient criterion extend to infinite paths composed of an infinite sequence of segments.

Following a segment with arbitrary precision is essentially a robust reachability problem. If the endpoint of the segment is robustly reachable from its starting point, then we can, for a given ε, define an convex set, where each point has distance at most ε to the segment, and that contains the $\varepsilon/2$ ball around the goal. We then run Algorithm 1.

This can be extended to piecewise smooth (continuously differentiable) paths that can be approximated arbitrarily closely by a (possibly infinite) sequence of segments, where the endpoint of each segment is reachable from its starting point. This is the case iff the derivation satisfies everywhere (where defined) the condition for robust reachability.

5. MINIMUM DWELL-TIME CONDITION

In this section, we consider an extension of robust reachability to robust reachability with or without dwell-time or discrete sampling. We assume w.l.o.g. that the minimal dwell-time or the sampling rate, respectively, is 1.

Theorem 11. *The robust reachability problem with dwell-time requirement is decidable for BMS where all rate vectors are positive.*

Proof. W.l.o.g assume that the starting state in $\mathbf{0}$ and the target state is $\overline{x}_t$. Notice that since all the rate vectors are positive, and every mode should be taken for at least 1 time-unit, there is a bound K such that the target state is not reachable is it is not reachable in K steps. (K is easy to compute.)

For robust reachability under bounded steps one can write a formula in first-order theory of reals. Now the decidability of the robust reachability with dwell-time requirement for BMS with positive rate vectors follows from the decidability of the first-order theory of reals. □

Theorem 12. *The reachability problem is* EXPTIME*-hard for MMS with dwell time requirements or discrete sampling.*

PROOF. We prove the result by a reduction from countdown games [10]. A countdown game is a tuple $\mathcal{G} = (N, T, (n_0, B_0))$, where N is a finite set of nodes, $T \subseteq N \times \mathbb{N}_{>0} \times N$ a set of transitions, and $(n_0, B_0) \in N \times \mathbb{N}_{>0}$ is the initial configuration. The states of a countdown game, also called its configurations, are $N \times \{0, 1, \ldots, B_0\}$.

From any configuration (n, B), Player 1 chooses a number $l \in \mathbb{N}_{>0}$ such that there exists a transition $(n, l, n') \in T$ with $l \leq C$. Among all the available transitions of the form (n, l, n'), Player 2 selects an appropriate transition $(n, l, n'') \in T$. The new configuration is then $(n'', C - l)$.

Player 1 wins when a configuration $(n, 0)$ is reached, and otherwise loses when a configuration (n, C) is reached where Player 1 cannot move. This is the case when, for all outgoing transitions $(n, l, n') \in T$, we have $l > C$. W.l.o.g., we assume that there are no transitions $(n, l, n) \in T$ for any $l \in \mathbb{N}_{>0}$.

We now translate this game into a sampled robust reachability problem, where the scheduler takes the role of Player 1, while the environment takes the role of Player 2.

The translation uses $|N| + 1$ variables, a variable B reflecting the remaining time budget and a variable n for each element $n \in N$. Being in state (n, C) in the countdown game is intuitively represented by $B = C$, $n = 1$, and $n' = 0$ for all states $n' \neq n$. The initial state is given by $B = B_0$, $n_0 = 1$, and $n = 0$ for all states $n \neq n_0$, i.e., by the state representing the initial configuration (n_0, B_0). The target is $\mathbf{0}$. The safety set is described by $n \in [-0.5, 1.5]$ for all $n \in N$ and $B \in [-0.5, B_0 + 1]$.

The rates Player 1 selects become the modes of our MMS. Thus, we have a mode l for each $l \in \mathbb{N}_{>0}$, for which a transition $(n, l, n') \in T$ exists. The selection of the concrete transition by Player 2 becomes the choice of the mode by the environment. We therefore have, for a given mode l, one rate vector for each transition $(n, l, n') \in T$, where the rates are $n = -1$, $n' = 1$, $B = -l$, and $n'' = 0$ for all $n'' \in N \smallsetminus \{n, n'\}$.

Before we describe how to translate (winning) strategies, we first note that, from each translation of a configuration, the scheduler cannot make a move of length ≥ 2. We first replace the target vertex by a the target region $B = 0$. For this target region, there is a simple 1:1 translation between the moves and states for the countdown game and the reachability game, where each move l of Player 1 in the countdown game corresponds to the move $(l, 1)$ of the scheduler, while every move (n, l, n') of Player 2 corresponds to the environment selecting the corresponding rate.

To return to normal reachability, we add, for each node $n \in N$, a mode n. This mode has only one rate, with $B = 0$, $n = -1$, and $n' = 0$ for all $n' \neq n$. Note that such a mode n can only be applied from states that encode (n, C), and it can only be applied with duration 1. Once such a mode is applied, no further mode (of either type) can be applied in the future, as one variable $n'' \in N$ would afterwards have the value -1.

Now, a winning for Player 1 corresponds to winning strategy of the scheduler that ends by applying such a mode. This closes the proof for discrete sampling.

To expand this to dwell time, we sharpen the bounds for the safety set to $n \in [-\varepsilon, 1 + \varepsilon]$ for all $n \in N$ and $B \in [-\varepsilon, B_0 + 1]$ for some $\varepsilon < \frac{1}{8B_0}$. Now, if Player 1 wins, then the scheduler wins with the same strategy as above. If Player 2 wins, Player 1 is stuck in $< B_0$ move pairs. When the environment mimics such a strategy (until Player 1 is stuck) then the game reaches a position, where each variable value is in a $2(B_0 - 1)\varepsilon < \frac{1}{4}$ range around the value it would have, had the scheduler played a duration of 1 for each move. Thus, the scheduler can, at most, play a "node mode" $n \in N$ once, but it cannot reduce the value of B without leaving the safety region. $\square$

For discrete sampled controllers, we can easily show inclusion in EXPTIME by exploring the complete state-space. To do this, we can proceed in two steps. In a first step, we expand all values in the problem setting to integers by multiplying every value with the least common multiple of all denominators. (Note that this is a polynomial time reduction.) Then we can be sure that all values are at integer points, and we can simply explore the complete state-space, which is exponential in the setting. As the lower bound is inherited from the previous proof, we get:

COROLLARY 13. *The robust reachability problem with discrete sampling is* EXPTIME*-complete.*

6. GENERALIZED MODELS

In this section we consider generalization of the BMS by adding structure to the model using Alur-Dill style [3] clock variables, i.e. variables with rate 1 in every mode. In the resulting model only clock variables can occur on the transitions where they can be compared against natural numbers or can be reset. All other non-clock variables will behave like BMS. We show that for BMS with clock the robust reachability problem is undecidable for BMS with 2 variables and 1 clock, and BMS with 1 variable and 2 clocks.

We prove the undecidability of this problem by giving a reduction from the halting problem for two-counter machines. Formally, a two-counter machine (Minsky machine) $\mathcal{A}$ is a tuple (L, C) where: $L = \{\ell_0, \ell_1, \ldots, \ell_n\}$ is the set of instructions. There is a distinguished terminal instruction ℓ_n called HALT. $C = \{c_1, c_2\}$ is the set of two *counters*; the instructions L are one of the following types:

1. (increment c) $\ell_i : c := c + 1$; goto ℓ_k,
2. (decrement c) $\ell_i : c := c - 1$; goto ℓ_k,
3. (zero-check c) ℓ_i : if $(c > 0)$ then goto ℓ_k else goto ℓ_m,
4. (Halt) ℓ_n : HALT.

where $c \in C$, $\ell_i, \ell_k, \ell_m \in L$.

A configuration of a two-counter machine is a tuple (ℓ, c, d) where $\ell \in L$ is an instruction, and c, d are natural numbers that specify the value of counters c_1 and c_2, respectively. The initial configuration is $(\ell_0, 0, 0)$. A run of a two-counter machine is a (finite or infinite) sequence of configurations $\langle k_0, k_1, \ldots \rangle$ where k_0 is the initial configuration, and the relation between subsequent configurations is governed by transitions between respective instructions. The run is a finite sequence if and only if the last configuration is the terminal instruction ℓ_n. Note that a two-counter machine has exactly one run starting from the initial configuration. The *halting problem* for a two-counter machine asks whether its unique run ends at the terminal instruction ℓ_n. It is well known that the halting problem for two-counter machines is undecidable.

THEOREM 14. *The robust reachability problem is undecidable for* BMS *with* 2 *variables and* 1 *clock.*

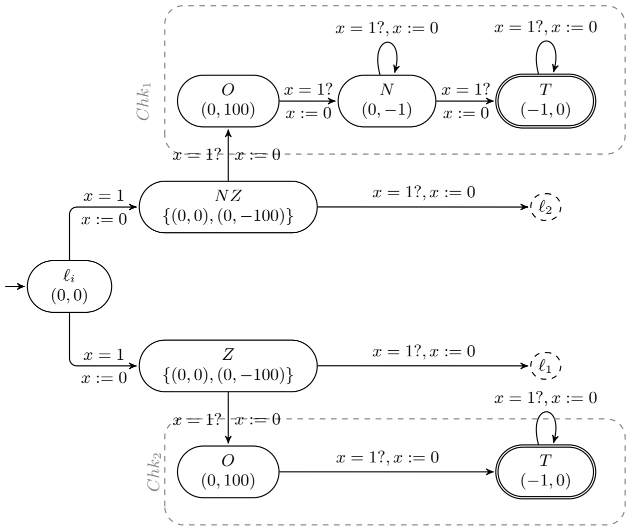


Figure 3: Simulation of Zero Check instruction

PROOF. For the sake of simplicity of presentation we prove the undecidability of the exact reachability problem. The proof can be adapted to robust reachability case. Given a Minsky machine we construct a structured BMS $\mathcal{H}$ with 2 variables and a single clock that is reset on every transition. The clock is used in a simple way just to ensue that at each mode exactly 1 unit of time is spent by the controller. We use two variables y and z to encode the values of the two counters c_1 and c_2, and one mode corresponding to each location of the Minsky machine. For each zero check instruction we further use five extra modes and a special target mode $\mathcal{T}$ depicted by a double circle. Our goal is to reach mode $\mathcal{T}$ with $y = z = 0$.

The simulation of the increment and decrement instruction is straightforward. In an increment c_1 location the rate is given by $(1, 0)$, while in decrement location the rate is give by $(-1, 0)$. Clock variables are used to ensure that exactly one time unit is spent in each such mode.

The Zero Check Instruction is simulated using the widget shown in Figure 3. The scheduler non-deterministically guesses if c_2 is zero or not, by going to one of the locations Z, NZ. The values of variables y, z remain unchanged. Assume that scheduler chose NZ. The environment can now allow the scheduler to continue his simulation by either giving the rate $(0, 0)$, or check his guess by giving the rate $(0, -100)$. If the rate $(0, 0)$ is obtained, the scheduler's best strategy is to goto ℓ_2, otherwise, the scheduler must go to $Chk1$. The first thing that happens in the gadget $Chk1$ is the variable z regaining its previous value by adding 100. If the scheduler's choice of c_2 being non-zero was incorrect, then when the location T is reached, we have $z = -1$. There is then no way to reach the target mode T with valuation $y = 0, z = 0$.

In a similar way, the environment can check if the scheduler guessed that the counter c_2 is zero, by giving the rate $(0, -100)$ at the location Z. In this case, the best strategy for scheduler is to goto the gadget $Chk2$. The first thing that happens in $Chk2$ is for variable z to regain its previous value by adding 100. If the guess of c_2 being 0 was correct, then the scheduler can reach T with $y = z = 0$. However, if the guess was wrong, scheduler can obtain $z = 0$, and will lose.

If the two counter machine halts, and the scheduler simulates all the instructions correctly, then it is possible to reach a mode $T \in \mathcal{T}$ with $y = z = 0$, or the mode $Halt$ is reached. It is straightforward to see that the location $Halt$ is reached iff the two counter machine halts and scheduler simulates all instructions correctly. From the $Halt$ mode we add an outgoing transition from where it is always possible for the scheduler to reach a mode $T \in \mathcal{T}$ with $y = z = 0$. The proof is now complete. $\square$

The proof of the following theorem is also via a reduction from the Minsky machines and is slightly more involved than the previous theorem. However, due to space limitation, we have moved the proof to the appendix.

THEOREM 15. *The robust reachability problem is undecidable for* BMS *with* 1 *variable and* 2 *clocks.*

Acknowledgments

We thank Rajeev Alur, Salar Moarref and Vojtech Forejt for the discussions related to some aspects of this work.

7. REFERENCES

[1] A.P. Aguiar and J.P. Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *Automatic Control*, 52(8):1362–1379, 2007.

[2] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-S. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, 1992.

[3] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[4] R. Alur, V. Forejt, S. Moarref, and A. Trivedi. Safe schedulability of bounded-rate multi-mode systems. In *HSCC*, pages 243–252, 2013.

[5] R. Alur, A. Trivedi, and D. Wojtczak. Optimal scheduling for constant-rate multi-mode systems. In *HSCC*, pages 75–84, 2012.

[6] E. Asarin, M. Oded, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *TCS*, 138:35–66, 1995.

[7] Michael S Branicky, Vivek S Borkar, and Sanjoy K Mitter. A unified framework for hybrid control: Model and optimal control theory. *Automatic Control*, 43(1):31–45, 1998.

[8] Luca De Alfaro and Thomas A Henzinger. Interface theories for component-based design. In *Embedded Software*, pages 148–165. Springer, 2001.

[9] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Comp. and Sys. Sciences*, 57:94–124, 1998.

[10] M. Jurdziński, J. Sproston, and F. Laroussinie. Model checking probabilistic timed automata with one or two clocks. *LMCS*, 4(3), 2008.

[11] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Robotics and Automation*, volume 1, pages 473–479. IEEE, 1999.

[12] J.L. Le Ny and G.J. Pappas. Sequential composition of robust controller specifications. In *Robotics and Automation*, pages 5190–5195, 2012.

APPENDIX
Proof of Theorem 15

We prove the undecidability by constructing a structured BMS $\mathcal{H}$ with 2 clocks and one variable that simulates the 2 counter machine. We prove that the scheduler has a winning strategy to reach $x \in B_\Delta(p)$ iff the two counter machine halts. Our construction of $\mathcal{H}$ is such that we have a gadget corresponding to each instruction in the two counter machine. We consider $p = 7$, and $0 < \Delta < 1$ as given. Modes in the target set $\mathcal{T}$ are denoted by a double circle.

Let the single variable be denoted z, and let x, y be the clocks. On entry into any gadget, the value of the variable z is $5 - \frac{1}{2^{c_1}3^{c_2}}$ where c_1, c_2 are the current values of the two counters, and the clocks x, y are zero.

1. Simulation of an increment instruction $\ell_i : c_1 := c_1 + 1$; goto ℓ_k.

   The gadget simulating the increment c_1 instruction can be seen in Figure 4. The gadget is entered with $z = 5 - \frac{1}{2^{c_1}3^{c_2}}, x = y = 0$. The locations in the gadget contain the name of the location as well as the rate (possibly a set of rates, or an interval of rates) of the variable z, as the case may be. Let us denote by *old* the value $\frac{1}{2^{c_1}3^{c_2}}$. A non-deterministic amount of time is spent at location ℓ_i. The ideal time to be spent here is $\frac{old}{2}$, so that z is updated from $5 - old$ to $5 - \frac{old}{2}$, reflecting the correct new counter values. y is reset on going to location A. A time of one unit is spent at location A. The value of x is unchanged during this process due to the self loop on A. There are three possible rates that the environment can give to the scheduler, namely 100, -100 or 0 at location A. The scheduler can go to any of the gadgets $C_>, C_<$ or to the location ℓ_k.

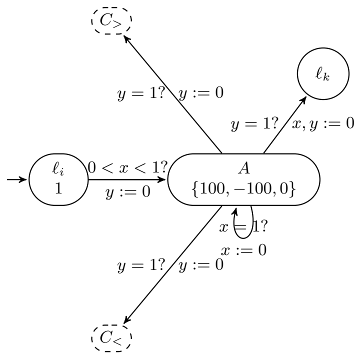


Figure 4: Simulation of increment c_1 instruction

   Assume that the time spent at ℓ_i is $\frac{old}{2} + \epsilon$ for some $\epsilon \geq 0$. In this case, $x = \frac{old}{2} + \epsilon, y = 0$ and $z = 5 - \frac{old}{2} + \epsilon$. The environment can force a check of the scheduler and catch his mistake, by choosing a rate of 100 at location A. This would make $z = 105 - \frac{old}{2} + \epsilon$. If the scheduler wants to win, he must reach a mode in $\mathcal{T}$, with the value of z in $B_\Delta(7)$. The best thing for the scheduler to do at this point is to choose $C_>$ as his next location, since it allows the value of z to come back to $5 - \frac{old}{2} + \epsilon$. If the scheduler chooses to go to $C_<$, he will be worse off, making z even bigger, and if he chooses ℓ_k, the environment can make sure that the scheduler never wins by choosing the rate 100 in all future gadgets.

Lets thus assume that the scheduler chooses to goto the gadget $C_>$ in Figure 5. On entry, we have $x = \frac{old}{2} + \epsilon$, $y = 0$ and $z = 105 - \frac{old}{2} + \epsilon$. At location O, the value of x remains unchanged, y grows to 1 and is reset, and z becomes $z = 5 - \frac{old}{2} + \epsilon$. At location B, a time $1 - \frac{old}{2} - \epsilon$ is spent, obtaining $x = 0, y = 1 - \frac{old}{2} - \epsilon$ and $z = 5 - \frac{old}{2} + \epsilon - 1(1 - \frac{old}{2} - \epsilon) = 4 + 2\epsilon$. If $\epsilon > 0$ and $2\epsilon > \Delta$, then the scheduler has already lost, since adding 3 more to z at location C does not help. Consider now the case that $\epsilon > 0$ and $\Delta - 2\epsilon = \kappa > 0$. At location D, a time of one unit is spent, and the environment can choose a rate as close to Δ as he wants : in particular, he can choose a rate that is larger than κ, making the value of $z = 4 + 2\epsilon + \kappa + \zeta$, for some $\zeta > 0$. This means the scheduler can never reach a point in the ball $B_\Delta(7)$, even after adding 3 to z at location C.

If $\epsilon = 0$, then irrespective of the rate $\kappa \in (0, \Delta)$ chosen by the environment, the value of z is $7 + \kappa \in B_\Delta(7)$, after adding 3 to z at location C. Thus, if the scheduler made no mistake, he reaches a point inside the chosen ball.

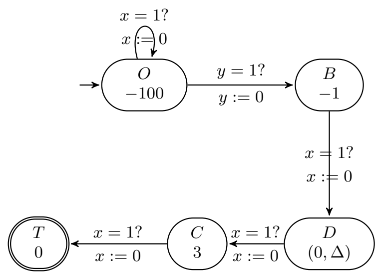


Figure 5: The gadget $C_>$

Now consider the case when the scheduler spends an amount of time $\frac{old}{2} - \epsilon$, for some $\epsilon \geq 0$ at location l_i in Figure 4. Then we have $x = \frac{old}{2} - \epsilon, y = 0$ and $z = 5 - \frac{old}{2} - \epsilon$. At location A in Figure 4, as seen above, the environment can assign any of the rates 100, -100 or 0 to the scheduler. If the environment wishes to catch the scheduler's mistake, a rate of -100 will be assigned. The scheduler, if he chooses to goto $C_>$ or *Go*, will surely lose, since the value of z will decrease further, and will never reach a value in $B_\Delta(7)$; likewise, if the scheduler chooses *Go*, the environment can forever give a rate of -100. The best choice for scheduler is therefore, to pick $C_<$. The gadget $C_<$ is given in Figure 6.

On entry to $C_<$, we have $x = \frac{old}{2} - \epsilon, y = 0$ and $z = -95 - \frac{old}{2} - \epsilon$. At location O, the value of x remains unchanged, y grows to 1 and is reset, and z becomes $z = 4 - \frac{old}{2} - \epsilon$. At location B, a time $1 - \frac{old}{2} + \epsilon$ is spent, obtaining $x = 0, y = 1 - \frac{old}{2} + \epsilon$ and $z = 5 - old$. A time $\frac{old}{2} - \epsilon$ is spent at location C, obtaining $z = 5 - old + 2(\frac{old}{2} - \epsilon) = 5 - 2\epsilon$.

At location E, a time of one unit is spent, and the environment can choose a rate in $(-\Delta, 0)$. Consider the case when $\epsilon > 0$ and $5 - 2\epsilon < 5 - \Delta$. In this case, scheduler has already lost the game, since spending one unit at location D will only give $z = 7 - 2\epsilon < 7 - \Delta$. However, if $7 - 2\epsilon > 7 - \Delta$, let $\kappa = \Delta - 2\epsilon > 0$.

Environment can then choose a rate $-\kappa - \zeta \in (-\Delta, 0)$, for $\zeta > 0$. Then $z = 7 - 2\epsilon - \kappa + \zeta = 7 - \Delta - \zeta < 7 - \Delta$. This would result in scheduler losing. However, if $\epsilon = 0$, then for any $-\kappa \in (-\Delta, 0)$, the value of z is $7 - \kappa \in B_\Delta(7)$.

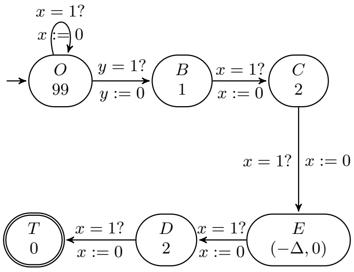


Figure 6: The gadget $C_<$

The only remaining case is when the scheduler indeed picks the correct delay of $\frac{old}{2}$ at location ℓ_i in Figure 4. In this case, as seen above, the rates 100, -100 chosen by the environment does not affect the scheduler. In both these cases, scheduler has a winning strategy of choosing to go to one of $C_<, C_>$ and reach a value of z in the chosen ball. If $\epsilon = 0$, and the environment picks the rate 0 at location A, then the best strategy for the scheduler is to select ℓ_k, which marks the continuation of the simulation of the two counter machine. As expected, we will indeed have on entry into l_k, $x = y = 0$ and $z = 5 - \frac{old}{2}$, marking the correct simulation of the increment c_1 instruction.

2. Simulation of a decrement instruction $\ell_i : c_1 := c_1 - 1$; goto ℓ_k.

   The construction of gadgets for the decrement instruction is similar to that of the increment instruction.

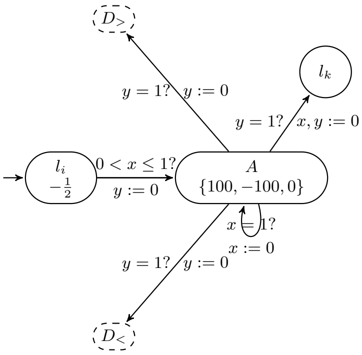


Figure 7: Simulation of decrement c_1 instruction

The ideal amount of time to be spent by scheduler at ℓ_i is $2old$. In this case, $z = 5 - 2old, x = 2old, y = 0$. Assume that the time spent at ℓ_i is $2old + \epsilon$, for some $\epsilon > 0$. Then we have $z = 5 - 2old - \frac{\epsilon}{2}$. At location A, the environment picks one of the 3 rates 100, -100, 0. If he wants to force a check on the environment, he picks the rate 100, making $z = 105 - 2old - \frac{\epsilon}{2}, x = 2old + \epsilon, y = 0$. As seen in the case of the increment gadget, the best strategy for the scheduler is to pick the gadget $D_>$.

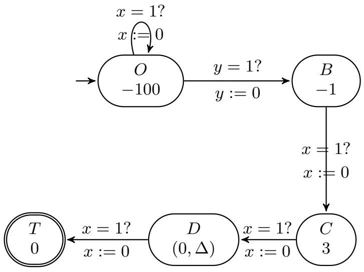


Figure 8: The gadget $D_>$

Entry into $D_>$ is made with $z = 105 - 2old - \frac{\epsilon}{2}$, $x = 2old + \epsilon, y = 0$. One unit of time is spent at O, obtaining $z = 5 - 2old - \frac{\epsilon}{2}$, $x = 2old + \epsilon, y = 0$. At B a time $1 - 2old - \epsilon$ is spent, obtaining $z = 5 - 2old - \frac{\epsilon}{2} - 1(1 - 2old - \epsilon) = 4 + \frac{\epsilon}{2}$. A time of one unit is spent at C obtaining $z = 7 + \frac{\epsilon}{2}$. Likewise, a time of one unit is spent at D, obtaining $z = 7 + \frac{\epsilon}{2} + \kappa$, where $\kappa \in (0, \Delta)$. If $\frac{\epsilon}{2} > \Delta$, then clearly, scheduler has already lost the game. If $\Delta - \frac{\epsilon}{2} > 0$, then $\kappa \in (0, \Delta)$ can be chosen such that $\kappa > \Delta - \frac{\epsilon}{2}$ such that the value of $z \notin B_\Delta(7)$. Note that if $\epsilon = 0$, this is not possible, and scheduler can indeed reach $z \in B_\Delta(7)$.

Now consider the case when scheduler spends a time $2old - \epsilon$, for some $\epsilon > 0$ at ℓ_i in Figure 7. Then we have $z = 5 - 2old + \frac{\epsilon}{2}$. Again, the environment can choose the rate -100 at location A, and the scheduler's best strategy is to enter gadget $D_<$. Entry into $D_<$ happens with $z = -95 - 2old + \frac{\epsilon}{2}, x = 2old - \epsilon, y = 0$. One unit of time is spent at O obtaining $z = 5 - 2old + \frac{\epsilon}{2}, x = 2old - \epsilon, y = 0$. A time $1 - 2old + \epsilon$ is spent at location B obtaining $z = 5 - 2old + \frac{\epsilon}{2} - 1(1 - 2old + \epsilon) = 4 - \frac{\epsilon}{2}$. One unit of time is spent at C obtaining $z = 7 - \frac{\epsilon}{2}$. Spending one unit at location D with a rate $-\kappa \in (-\Delta, 0)$ gives $z = 7 - \frac{\epsilon}{2} - \kappa$. If $\frac{\epsilon}{2} > \Delta$, then the scheduler has already lost. If $\Delta - \frac{\epsilon}{2} > 0$, then the environment can always choose $-\kappa \in (-\Delta, 0)$ such that $z = 7 - \frac{\epsilon}{2} - \kappa < 7 - \Delta$. Clearly, if $\epsilon = 0$, this is not possible. and scheduler wins.

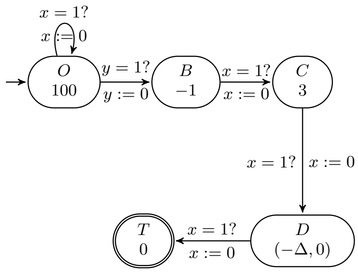


Figure 9: The gadget $D_<$

3. Zero Check Instruction. ℓ_i: if $c_2 = 0$ goto ℓ_1 else goto ℓ_2.

   Figure 10 describes the gadget for zero check of counter c_2. No time is spent at location ℓ_i, and the scheduler makes a guess about the value of c_2. If he guesses that c_2 is zero, then he will choose the location Z. The environment can either allow the scheduler to go ahead with the simulation by choosing a rate 0, or could verify the correctness of scheduler's guess by choosing a rate 100. One unit of time has to be spent at the location Z. Thus, if the scheduler decides to verify and chooses the rate 100, the value of z will be $105 - old$. The environment will check if $old = \frac{1}{2^{c_1}}$, for some $c_1 \geq 0$. If the environment chooses 100, the best strategy for the scheduler is to choose the gadget $Z?$. Going to ℓ_1 does not help the scheduler to win, since the environment can pick the rate 100 in all future choice locations, ensuring that the scheduler cannot win.

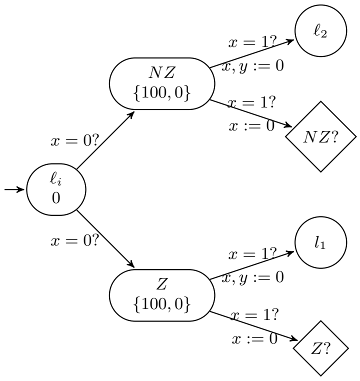


Figure 10: Zero Check

The gadget $Z?$ given in figure 11 is a check gadget which checks if $old = \frac{1}{2^{c_1}}$, for some $c_1 \geq 0$. The gadget $Z?$ is entered with $z = 105 - old$, $x = y = 0$. A time of unit is spent at location L, obtaining $z = 5 - old$ and $x, y = 0$. If indeed $c_2 = 0$, and if in addition, $c_1 = 0$, then $old = 1$ and $z = 4$. In this case, the scheduler can go to the location F, spend a unit of time at F obtaining $z = 7$. This leads to the location G, where the environment can pick any rate in $(0, \Delta)$. One unit of time is spent in G, and in this case, we reach the mode T with $z = 7 + \kappa \in B_\Delta(7)$, for $\kappa \in (0, \Delta)$. Clearly, the scheduler wins here since his guess about c_2 being zero was correct.

In case $old = \frac{1}{2^{c_1}}$ for $c_1 > 0$, then from location M, the scheduler cannot win by choosing location F as the next location, since the value of z on entry into G will be $8 - old$, where $old = \frac{1}{2^{c_1}}$ for $c_1 > 0$. If $8 - old > 7 + \Delta$, then the scheduler has already lost. If $8 - old \leq 7 + \Delta$, let $8 - old = (7 + \Delta) - p$, for some $p \geq 0$. Then $p = \Delta + old - 1 < \Delta$, since $old \leq \frac{1}{2}$. Thus, the environment can pick a rate $p + \zeta \in (0, \Delta)$ such that $z = 8 - old + p + \zeta > 7 + \Delta$.

Thus, if $c_1 > 0$, the best strategy for scheduler is to goto location C. The subgraph consisting of locations C, D and gadgets $D_<$ and $D_>$ simulates the decrement c_1 instruction. The ideal time to be spent at C is $2old$ so that the value of c_1 is decremented by one. At location D, the environment can choose a rate 0 (in which case, scheduler will go back to location C) or a rate 100 (in which case scheduler will go to $D_>$) or a rate -100 (in which case, scheduler will go to $D_<$). In the case scheduler goes back to C, the new value of z is $5 - 2old, x = y = 0$. The ideal time to be spent at C now is $4old$, and so on. At some point of time when $c_1 = 0$,

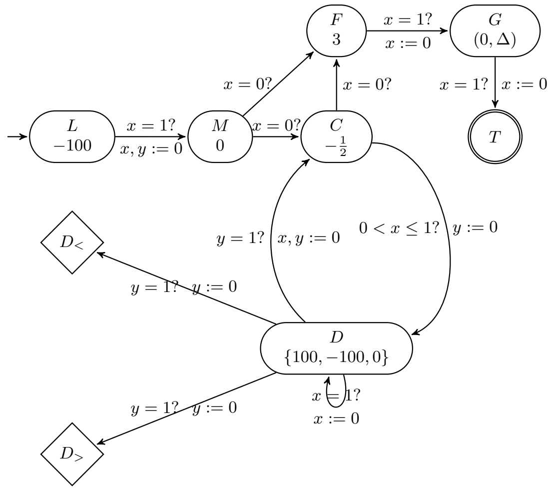


Figure 11: The gadget $Z?$

we will obtain $z = 4$. At this point, the scheduler can take the transition to F from C, and as seen above can reach T with $z \in B_\Delta(7)$. If the scheduler goes to F from C when $z = 5 - old$ for some $0 < old < 1$, then as seen above, on entry into G, $z = 8 - old$, and the environment has a choice of rate in $(0, \Delta)$ such that scheduler loses.

The gadget $NZ?$ is given in Figure 12. This is entered into when the environment chooses a rate of 100 at location NZ in Figure 10. The idea is to verify that indeed c_2 is non-zero. Scheduler has to go through the locations C, E atleast once so that c_2 is decremented atleast once (hence, $c_2 \neq 0$). The time elapse at C must be $3old$, so that $z = 5 - old - \frac{2}{3}(3old) = 5 - 3old$, decrementing c_2. The gadgets $E_<$ and $E_>$ can be designed similar to the gadgets $D_<$ and $D_>$ to catch the errors of the scheduler when the time elapse is $3old + \epsilon$ and $3old - \epsilon$, $\epsilon > 0$. The scheduler must visit the C, E loop c_2 times (provided the environment gives rate 0 at location E everytime). When the rate 0 is given at location E, scheduler can move to location J when c_2 becomes 0. If c_1 is zero, then we get $z = 4$ at the end of the C, E loop. Then from J, scheduler can go to location F spending no time at J, and reach T with $z \in B_\Delta(7)$. However, if $c_1 > 0$, then scheduler visits the J, K loop until $c_1 = 0$ (provided the environment gives a rate 0 at location K). When $c_1 = 0$, the scheduler can move from J to F, and reach the target with $z \in B_\Delta(7)$.

4. The Halt location : The location labeled $Halt$ has rate 1. The scheduler will reach here iff the two counter machine halts, and when the scheduler has simulated all the instructions correctly. The value of z will be $5 - old$, where $old = \frac{1}{2^{c_1} 3^{c_2}}$, for $c_1, c_2 \geq 0$. A non-deterministic amount of time can be spent by the scheduler here so that z will lie in $B_\Delta(7)$.

It can be proved that the scheduler has a winning strategy to reach $z \in B_\Delta(7)$) iff the two counter machine halts.

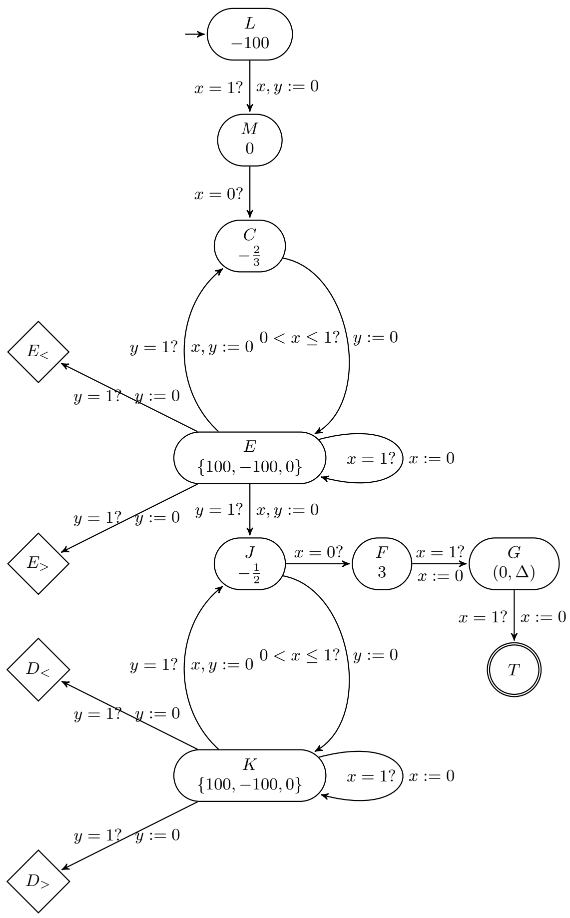


Figure 12: The gadget $NZ?$